TRIROUTE: ROUTING TOOL-CALL FAILURES ACROSS RESAMPLING, STOPPING, AND EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

The structure of a failed tool call can reverse the value of its next recovery action: diffuse malformed outputs favor fresh samples, concentrated syntax-valid faults favor local edits, and low-separation states can make stopping preferable. *TriRoute* operationalizes this geometry with a frozen six-feature score that selects among Best-of-5, Keep, and a radius-three TriEdit search. To distinguish the value of the resulting policy from a stronger repair arm, routed and greedy policies share the generator, verifier, prompts, seeds, five-template edit library, and a hard ceiling of six verifier evaluations. With GPT-4o, TriRoute improves over always invoking that identical edit library by 0.73–0.76 percentage points on APIBench-Tool, BFCL v3, and API-Bank, with paired intervals excluding zero, while using 0.23–0.25 fewer verifier evaluations per recovery episode. Against schema-aware validator-feedback retry, it reaches 81.59% on BFCL v3 and 75.23% on API-Bank—gains of 1.05 and 1.21 points—and reduces recovery-episode calls from $5.62/5.58$ to $4.91/4.89$. Score strata and failure slices exhibit the predicted shift from resampling to editing; without generator-specific fitting, routed-minus-validator intervals exclude zero in all nine cells spanning GPT-4o, Llama-3.1-70B, and DeepSeek-32B, while same-library point estimates are positive in all nine and six intervals exclude zero. In verifier-rich structured generation, assigning an existing recovery repertoire from the observed failure state raises success while reducing verification demand, without enlarging the repair library.

1 INTRODUCTION

Two tool calls can receive similarly imperfect verifier scores and still require opposite responses. An object missing one required field preserves nearly all of the intended arguments, so a local edit can exploit its structure. A malformed object with distributed violations provides no comparable neighborhood, so fresh sampling is more promising. Between these cases, measured action values are weakly separated and stopping remains competitive. The useful unit of inference-time control is therefore a *failure state paired with a recovery operator*, not merely another sample or search step.

End-to-end repair comparisons entangle the quality of an operator with the policy that invokes it. Adding a learned editor, changing its trigger, and granting extra evaluations can all raise final success, but they answer different design questions. We target the policy question with a same-library contrast: routed and greedy policies receive the same generator, verifier, prompts, five edit templates, seeds, and hard $B{=}6$ verifier-evaluation ceiling. Their difference is the complete invocation policy—whether a failed request is kept, independently resampled, or sent to the shared edit arm—yielding a controlled estimate for replacing unconditional editing with state-dependent triage.

TriRoute realizes this comparison as an ordered partition of verifier-visible failures. A frozen logistic score combines initial pass rate, distance to perfect verification, the number and concentration of failed checks, syntax validity, and candidate length. Two thresholds are selected jointly against the realized outcomes of Keep, Best-of-5, and TriEdit rather than by fitting independent triggers. The low-score region buys independent samples, the high-score region searches a radius-three argument neighborhood, and the validation objective assigns the low-separation center to Keep. Coefficients and thresholds are fit on MBPP-Plus recovery attempts and then frozen before every tool-call and cross-model evaluation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The direct same-library comparison is positive on all three GPT-4o benchmarks: $+0.76$ points on APIBench-Tool, $+0.73$ on BFCL v3, and $+0.76$ on API-Bank, with paired intervals excluding zero and 0.23–0.25 fewer verifier evaluations per recovery episode. Against validator-feedback retry, TriRoute gains 1.05 and 1.21 points on externally authored BFCL v3 and API-Bank while saving 0.71 and 0.69 recovery-episode evaluations. The unchanged policy also has positive routed-minus-validator means with intervals excluding zero in all nine model–benchmark cells. Independently measured score strata on APIBench-Tool and BFCL v3 exhibit the same low-to-high transition from resampling advantage to editing advantage.

The paper contributes a controlled estimate of routed-policy value under a fixed repair library and shared verifier ceiling, a three-region formulation that connects observable failure structure to heterogeneous recovery actions, and a frozen-policy evaluation across external benchmarks, three generators, validation-tuned heuristics, and altered operator coverage. Together, these results establish that inference-time allocation for structured calls should decide *which* recovery operator receives the remaining verification budget, not only how deeply one operator runs.

2 Related Work

2.1 Algorithm selection and inference routing

Algorithm selection maps observable instance features to one member of a portfolio (Rice, 1976). Modern LLM cascades and routers apply this idea to model choice or escalation under quality–cost trade-offs (Chen et al., 2024a; Ong et al., 2025). TriRoute instead acts *after* an initial candidate has exposed verifier diagnostics and chooses among heterogeneous recovery operators executed by the same generator. This distinction motivates both the failure-state representation and the same-library control: model-routing gains do not determine whether a tool-call failure should be kept, independently resampled, or edited in place.

2.2 Selective computation and test-time allocation

Verifier selection, self-consistency, tree search, and repeated sampling allocate additional inference to promising candidates (Cobbe et al., 2021; Lightman et al., 2024; Wang et al., 2023; Yao et al., 2023; Brown et al., 2024; Snell et al., 2025). These methods primarily vary depth or breadth within one action family. Failure-state routing compares the conditional value of different post-failure actions, including Keep as an explicit decision. A common hard ceiling and identical local-edit arm focus the comparison on the value of replacing unconditional repair with the three-action routed policy.

2.3 Structured tool generation and validator feedback

Toolformer, ToolLLM, API-Bank, RestGPT, and BFCL establish realistic tool-use tasks and evaluation interfaces (Schick et al., 2023; Qin et al., 2024; Li et al., 2023; Song et al., 2023; Patil et al., 2025). Grammar-constrained decoding, PICARD-style incremental constraints, and structured outputs strengthen syntactic validity (Geng et al., 2023; Scholak et al., 2021), while validator-feedback retry exposes schema errors to a new model call. They provide generation and retry baselines for our setting. TriRoute instead chooses a post-failure action from external verifier diagnostics while generation, validation, and the edit operators remain fixed.

2.4 Post-failure routing and invocation repair

Self-Refine, Self-Debug, Reflexion, and automated program-repair systems improve a selected candidate through critique, regeneration, or edit search (Madaan et al., 2023; Chen et al., 2024b; Shinn et al., 2023; Le Goues et al., 2019; Xia et al., 2023; Olausson et al., 2024). Invocation Refiner trains a separate reinforcement-learned post-processor to rewrite outputs from frozen upstream models and evaluates it on BFCL and API-Bank (Jiao et al., 2026); its contribution is a stronger reusable repair module. The subsequently withdrawn CodeRescue preprint describes routing failed coding-agent attempts among reflection, replanning, and model escalation and calibrating a cost frontier with conformal risk control (He et al., 2026). TriRoute addresses the complementary controlled question for tool-call arguments: its actions use one generator, its local edit capability is identical in routed

and greedy arms, Keep is explicit, and the resource contract is a hard verifier-evaluation ceiling rather than a deployment-time monetary frontier. Thus the closest systems differ along the repair capability, action space, and budget object, while Table 1 targets the value of the complete score-dependent invocation policy under a fixed edit library.

3 PROBLEM FORMULATION

Given an input x, a frozen generator $\mathcal{M}$ produces y_0. The external verifier $V : \mathcal{Y} \to \{0,1\}^m$ returns a pass vector, with normalized score $s(y) = \|V(y)\|_1/m$. The initial evaluation $V(y_0)$ is computed once, charged once, cached for routing, and passed to local search.

3.1 ACTION SET AND ACCOUNTING

The deployment menu contains KEEP, BEST-OF-5, and TRIEDIT. Every policy first pays for and caches $V(y_0)$. If $V(y_0)$ is perfect, every policy returns after that evaluation and no recovery episode is opened. Otherwise, Keep stops at the cached call, Best-of-5 evaluates four fresh samples in addition to y_0, and TriEdit may execute at most $K{=}5$ post-initial edit evaluations. Validator retry is bounded by the same $B{=}6$ total, whereas uncapped Best-of-8 appears only as secondary cost context.

Let $\mathcal{F} = \{(x,\omega) : s(y_0) < 1\}$ denote the set of recovery-eligible instance–seed attempts. Success means average over all attempts, including perfect initial candidates; every reported call, latency, token, and cost mean averages the charged ledger over $\mathcal{F}$. This common denominator explains why fixed Best-of-5 rows equal 5.00 even though perfect initial candidates exit after one evaluation. K/B/R summaries later in the paper aggregate only recovery decisions to one modal label per benchmark instance; they describe the partition but are not mixture weights for reconstructing attempt-level resource means.

3.2 OBJECTIVE AND ESTIMAND

For action a, let $\mathrm{Succ}_a(x,\omega) \in \{0,1\}$ and $\mathrm{Calls}_a(x,\omega)$ denote final task success and charged verifier evaluations for one instance–seed attempt. Policies optimize success lexicographically with recovery demand: first maximize $\mathbb{E}[\mathrm{Succ}_\pi(x,\omega)]$ subject to $\mathrm{Calls}_\pi(x,\omega) \le B$ for every attempt, then minimize $\mathbb{E}[\mathrm{Calls}_\pi(x,\omega) \mid \mathcal{F}]$ among validation-success-tied policies. The estimand is

$$\Delta_{\mathrm{policy}} = \mathbb{E}[\mathrm{Succ}_{\mathrm{TriRoute}}(x,\omega) - \mathrm{Succ}_{\text{greedy-TriEdit}}(x,\omega)],$$

where generator, verifier, prompts, edit menu, seeds, and hard cap are identical. This quantity removes edit-library variation from the routed-versus-greedy comparison and measures the value of the complete routed policy; its action mixture, stopping, resampling, and realized calls are policy outcomes rather than separately identified causal components.

3.3 ORACLE REGRET

A deployment policy maps $(x, y_0, V(y_0))$ to one of the three actions. Let

$$a^\star(x,\omega) = \arg\max_{a \in \{\text{KEEP},\text{BEST-OF-5},\text{TRIEDIT}\}} (\mathrm{Succ}_a(x,\omega), -\mathrm{Calls}_a(x,\omega))$$

denote the per-attempt oracle under the same menu, candidate draws, and accounting. Regret is $100\,\mathbb{E}[\mathrm{Succ}_{a^\star}(x,\omega) - \mathrm{Succ}_\pi(x,\omega)]$ percentage points. We define $\Delta_{\max} = 100(\mathbb{E}[\mathrm{Succ}_{a^\star}(x,\omega)] - \max_a \mathbb{E}[\mathrm{Succ}_a(x,\omega)])$, the oracle's advantage over the best uniform action.

3.4 FAILURE-STATE ACTION-VALUE GEOMETRY

Let $z = \phi(x, y_0, V(y_0))$ denote the verifier-visible failure state and define

$$Q_a(z) = (\Pr[\mathrm{Succ}_a = 1 \mid z], -\mathbb{E}[\mathrm{Calls}_a \mid z])\,.$$

The optimal triage region for action a contains states where $Q_a(z)$ is lexicographically maximal. This view replaces a scalar notion of repairability with relative action value: two candidates with similar verifier scores can favor different actions when one has concentrated, executable faults and the other has distributed violations. TriRoute estimates an ordered projection of these regions; Figure 1 descriptively examines the resulting low-score resample advantage, low-separation center, and high-score edit advantage on held-out requests.

3.5 STRUCTURED EDIT NEIGHBORHOOD

For the fixed operator set $\mathcal{O} = \{o_1, \ldots, o_p\}$ and an incumbent candidate y, define the one-expansion neighborhood

$$\mathcal{N}_r(y) = \{o_{i_1} \circ \cdots \circ o_{i_k}(y) : k \le r,\ o_{i_j} \in \mathcal{O}\}.$$

We use r=3. At every evaluated expansion, TriEdit composes at most three compatible operators from its current incumbent; when an expansion improves the verifier score, that candidate becomes the next incumbent. Consequently, a successful trajectory can contain more than three accepted edits relative to y_0, while every evaluated proposal remains in $\mathcal{N}_3(y^\star)$ for the incumbent $y^\star$ at that expansion. The same rolling neighborhood and update rule are used by routed and greedy repair.

4 METHOD

TriRoute implements a frozen partition of verifier-visible failure states under a shared resource constraint. TriEdit denotes the task-compatible local-edit arm; every tool-call evaluation instantiates it with the same fixed five-template menu. Greedy TriEdit invokes Algorithm 1 for every imperfect initial candidate, whereas TriRoute invokes it only above the high threshold and otherwise keeps or resamples. Freezing the scoring rule, thresholds, and tool-call menu before every reported test evaluation makes this an end-to-end comparison between a score-dependent three-action policy and unconditional use of the identical edit arm.

4.1 FIXED TRUST-REGION EDIT ARM

TriEdit transforms the current argument object rather than prompting a full rewrite. Its shared library contains `add-required-field`, `fix-type-coerce`, `remove-extra-field`, `fix-enum-value`, and `normalize-path`. The templates are mined once from held-out Tool-Bench validation diffs by canonicalizing touched schema paths and edit types, retaining support at least four and mean verifier gain at least 0.08, and removing benchmark-specific tool names and literals. APIBench-Tool, BFCL v3, API-Bank, τ-bench, and NexusRaven v2 all use this unchanged menu. Invalid and duplicate candidates are rejected before a verifier call; a valid composed edit is charged once when executed.

4.2 BUDGETED LOCAL EDIT SEARCH

Operator selection is a per-request low-budget bandit. Each operator o has a Beta posterior (α_o, β_o) recording whether its composed proposals improve the current verifier score. Thompson samples rank operators, up to r compatible edits are composed, and valid unseen candidates consume the next available evaluation.

The posterior tracks operator usefulness instead of token positions, concentrating up to five post-initial evaluations on edits that improve the current request. Cached v_0 is never recomputed, so Algorithm 1 and every baseline share the same initial charge and respect the six-evaluation ceiling in Table 5.

4.3 FROZEN FAILURE-STATE TRIAGE RULE

4.3.1 REPAIR-LOCALITY SCORE

Let $v = V(y_0)$ contain m binary checks. The six features are pass rate $s = \|v\|_1/m$, distance $1 - s$, failed-check count $m - \|v\|_1$, fail-cluster count, candidate length, and syntax validity. A fail cluster groups failed predicates attached to the same schema path or semantic post-condition identifier; candidate length is measured in generator-tokenizer units. For continuous feature j, the fitted input is $\tilde{\phi}_j = (\phi_j - \mu_j)/\sigma_j$ using router-training statistics, while syntax validity remains binary. The affine redundancy between s and $1 - s$ is retained in the regularized fit but adds no representational capacity. On each validation attempt, all three actions are executed under the common accounting rule with a task-compatible local-edit arm. The binary target is one exactly when local editing is lexicographically optimal and zero otherwise. Temperature scaling yields

$$h = \sigma\Big((w^\top \tilde{\phi} + b)/T\Big),$$

an ordered score for concentrated local repairability. High scores identify syntax-valid near misses for which local edits have the highest value; low scores collect malformed or diffuse failures, while intermediate scores capture states with weak separation among available actions.

4.3.2 JOINT THREE-ACTION THRESHOLD SELECTION

The router is fit once on 2,000 GPT-4o MBPP-Plus validation attempts (Liu et al., 2023; Austin et al., 2021) from 250 problems and eight attempts per problem. Five-fold cross-validation keeps all attempts from a problem in one fold. The pair $(\tau_{\text{low}}, \tau_{\text{high}}) = (0.30, 0.70)$ is selected jointly by grid search over the realized outcomes of all three actions under the success-first, conditional-call objective in Section 3; this three-action search, rather than the binary label alone, determines the Best-of-5 and Keep regions. The resulting standardizers, coefficients, temperature, and thresholds are frozen before tool-call evaluation. At inference, a perfect initial candidate exits immediately; otherwise $h < 0.30$ selects Best-of-5, $0.30 \leq h \leq 0.70$ selects Keep, and $h > 0.70$ selects TriEdit, so equality at either threshold deterministically favors Keep. No generator logits, hidden states, benchmark-specific feature additions, coefficient updates, or threshold updates enter the policy.

4.3.3 CALIBRATION AND PARTITION QUALITY

Temperature scaling is fitted to problem-held-out scores before threshold freezing. ECE for the *local-edit-optimal label*, rather than final task success, is computed after scaling on all 2,000 out-of-fold attempts with 15 equal-frequency bins and equals 0.029 (Guo et al., 2017). On that MBPP-Plus population, TriRoute has 0.61 percentage-point oracle regret, 0.186 Keep mass, and $\Delta_{\max} = 2.81$ points. Appendix A7 reports the predeclared cross-model transfer criteria beside the corresponding measurements.

5 EXPERIMENTS

The experiments answer three questions in sequence. *Q1:* Does TriRoute improve success under the same verifier ceiling as strong retry, constrained-generation, and repair baselines? *Q2:* Does the score-dependent policy improve over unconditional use of the identical TriEdit library? *Q3:* Do the frozen partition and its action-value ordering persist across generators, benchmarks, heuristics, and operator coverage?

5.1 SETUP AND EVALUATION PROTOCOL

5.1.1 MODELS AND INFRASTRUCTURE

The primary generator is Azure OpenAI's August 6, 2024 GPT-4o checkpoint, gpt-4o-2024-08-06 (OpenAI, 2024), queried with temperature 0.7, top-p 0.95, `max_tokens`=2048, and one retry on HTTP 5xx. Cross-model runs use Llama-3.1-70B-Instruct (Grattafiori et al., 2024) and DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025), served by PyTorch 2.4.1, vLLM 0.6.3, and FlashAttention 2.6.3 with tensor parallelism four on one 4×NVIDIA H100 80GB SXM node (CUDA 12.4, driver 550.90.07, NCCL 2.22.3, Ubuntu 22.04.4). Routing, operator execution, and verification run on an 80-core AMD EPYC 9654 host with 512 GB DDR5-4800; wall-clock measurements use this host.

5.1.2 BENCHMARKS AND METRICS

The evaluated BFCL v3 (Patil et al., 2025) subset contains 638 instances; API-Bank (Li et al., 2023) contributes 753 call-level instances from 314 dialogues. APIBench-Tool is a controlled ToolBench-derived study with 384 single-tool instances from 12 held-out tool classes, verified by JSON Schema Draft 2020-12 and semantic post-conditions (Qin et al., 2024). The evaluated NexusRaven v2 (Nexusflow.ai team, 2023) set contains 318 instances, and the predeclared τ-bench (Yao et al., 2025) support run has 165. Task success requires both argument validity and semantic completion. Success averages all attempts; verifier calls and the APIBench-Tool wall-clock, token, and dollar measurements average the shared set $\mathcal{F}$ of recovery episodes defined in Table 5.

The controlled evaluation treats each benchmark's executable checker as the recovery-time verifier: its parse, schema, execution, and post-condition failures are converted to the common pass-vector

Table 1: Same-library routed-policy estimate under $B{=}6$. Both columns use the frozen five-template menu; greedy always invokes TriEdit after failure, whereas TriRoute selects Keep, Best-of-5, or TriEdit. CIs resample instances, randomization p-values use per-instance ten-seed mean differences, W/T/L reports greedy wins / ties / TriRoute wins, and calls average recovery episodes.

Benchmark	Greedy	TriRoute	Δ [95% CI]	p	W/T/L	Calls
APIBench-Tool	78.58 ± 0.52	79.34 ± 0.51	$+0.76$ [$+0.31, +1.21$]	0.0021	222/3367/251	5.18/4.93
BFCL v3	80.86 ± 0.44	81.59 ± 0.42	$+0.73$ [$+0.28, +1.18$]	0.0016	291/5751/338	5.14/4.91
API-Bank	74.47 ± 0.48	75.23 ± 0.47	$+0.76$ [$+0.29, +1.23$]	0.0014	352/6769/409	5.12/4.89

Table 2: External results with Azure `gpt-4o-2024-08-06`. Success is mean percentage $\pm$ one seed standard error over all attempts. Δ is TriRoute minus validator-feedback retry; CIs are paired instance-bootstrap intervals. Calls are validator/TriRoute means per recovery episode.

Benchmark	n	Validator	Greedy	TriRoute	Δ [95% CI]	Calls
BFCL v3	638	80.54 ± 0.45	80.86 ± 0.44	81.59 ± 0.42	$+1.05$ [$+0.53$,$+1.57$]	5.62/4.91
API-Bank	753	74.02 ± 0.49	74.47 ± 0.48	75.23 ± 0.47	$+1.21$ [$+0.64$,$+1.78$]	5.58/4.89

features before an action is chosen. Official task success is evaluated only after the capped recovery sequence. For multi-turn τ-bench, routing occurs at a tool-call emission while the conversation prefix and environment state remain fixed across actions; the selected call is then returned to the native interaction loop.

5.1.3 Comparators and matching

All decision-critical rows share prompts, candidate temperature, verifier, seed grid, cached initial evaluation, recovery-episode denominator, and hard $B{=}6$ ceiling. The default comparator is schema-aware validator-feedback retry; Outlines-0.0.46 constrained decoding and strict structured output provide additional schema-aware baselines. Agentless-Repair, a localized tool-argument adaptation of Agentless (Xia et al., 2025), is a generic capped repair baseline, Best-of-5 is fixed-schedule resampling under the shared ceiling, and greedy TriEdit is the direct same-library control. Best-of-8 is uncapped cost context. Full results for Self-Debug, Reflexion, LATS, and Tree-of-Thoughts (Chen et al., 2024b; Shinn et al., 2023; Zhou et al., 2024; Yao et al., 2023) appear in Appendix A12.

5.1.4 Statistical protocol

All means and standard errors use seeds $\{17, 23, 29, 31, 37, 41, 43, 47, 53, 59\}$. A benchmark instance is the inferential unit: paired bootstrap intervals resample instances $10{,}000$ times while retaining each instance's ten-seed outcome vector. Corrected paired tests use the continuous per-instance mean success difference across seeds in a two-sided randomization test with 10^6 sign permutations; instance-seed win/tie/loss counts and mechanism slices are descriptive. This separates benchmark instances, stochastic seeds, and the $n{\times}10$ executed attempts.

5.2 Primary Outcomes and Policy Attribution

Table 1 is the central attribution test: it holds the local-edit menu fixed and compares the complete routed policy with unconditional editing. Table 2 provides the complementary practical comparison against schema-aware validator retry on two externally authored benchmarks.

Greedy repair is effectively tied with validator-feedback retry on APIBench-Tool (78.58 versus 78.56) and exceeds it on BFCL v3 and API-Bank. TriRoute improves on greedy in all three cases by 0.73–0.76 points and uses 0.23–0.25 fewer recovery-episode calls. With edit capability fixed, these paired results estimate the benefit of replacing unconditional editing with the frozen three-action policy.

Table 3: TriRoute on externally authored benchmarks with the unchanged router, thresholds, prompts, edit menu, and B=6 verifier ceiling. Calls average recovery episodes. Δ_V and Δ_G compare validator retry and same-library greedy repair. Full method rows, W/T/L counts, and corrected paired tests are in Appendix A7.

Model	Benchmark	Success	Calls	Δ_V [95% CI]	Δ_G [95% CI]
GPT-4o	BFCL	81.59	4.91	+1.05 [+0.53,+1.57]	+0.73 [+0.28, +1.18]
	API-Bank	75.23	4.89	+1.21 [+0.64,+1.78]	+0.76 [+0.29, +1.23]
	NexusRaven	74.38	4.92	+1.10 [+0.48, +1.72]	+0.71 [+0.26, +1.16]
Llama-3.1-70B	BFCL	74.82	4.98	+0.85 [+0.26,+1.43]	+0.63 [+0.08,+1.17]
	API-Bank	68.41	4.96	+0.86 [+0.21,+1.51]	+0.57 [−0.02,+1.16]
	NexusRaven	66.72	4.99	+1.16 [+0.39,+1.91]	+0.81 [+0.11,+1.50]
DeepSeek-32B	BFCL	76.28	4.95	+0.76 [+0.17,+1.34]	+0.49 [−0.06,+1.03]
	API-Bank	70.16	4.94	+0.96 [+0.30,+1.61]	+0.64 [+0.05,+1.22]
	NexusRaven	68.37	4.97	+0.80 [+0.09,+1.50]	+0.51 [−0.15,+1.16]

Table 4: Decision-focused APIBench-Tool results on 384 held-out instances. Success averages all attempts. Calls, tokens, USD, and wall-clock average recovery episodes; wall-clock is relative to their single-pass initial evaluation. All capped rows use the B=6 verifier ceiling.

Method	Success (%)	Calls	Wall (×)	kTok/episode	USD/episode
Best-of-5	76.22 ± 0.60	5.00	4.55	0.791	0.00771
Agentless-Repair	78.02 ± 0.56	5.71	5.40	0.987	0.00961
Strict structured output	77.48 ± 0.55	5.00	4.45	0.785	0.00766
Outlines-0.0.46 GBNF	77.91 ± 0.53	5.00	4.68	0.814	0.00792
Validator-feedback retry	78.56 ± 0.54	5.63	5.25	0.967	0.00943
Same-library greedy TriEdit	78.58 ± 0.52	5.18	4.47	0.802	0.00782
TriRoute	79.34 ± 0.51	4.93	4.20	0.728	0.00708
Best-of-8 (uncapped context)	76.40 ± 0.58	8.00	7.30	1.267	0.01234

5.3 Cross-Model Shared-Ceiling Results

The routed-minus-validator mean is positive in every model–benchmark cell, and all nine paired intervals exclude zero. The same-library point estimate is also positive in all nine cells; six intervals exclude zero, while Llama-3.1-70B on API-Bank and DeepSeek-32B on BFCL and NexusRaven include zero. The unchanged direction across open-weight backends supports transfer of the partition, while the interval pattern localizes the strongest same-library evidence to GPT-4o and six of the nine cross-model cells.

5.3.1 Predeclared τ-bench corroboration

Under the unchanged policy, TriRoute reaches $75.14 \pm 0.45\%$, versus 74.29% for validator retry, with 4.91 calls and paired CI $[+0.32, +1.38]$; the corrected randomization test gives $p = 0.0019$. Appendix A5 reports the full protocol, and Appendix A6 reports the 74.38% NexusRaven result.

5.4 Controlled Recovery Study on APIBench-Tool

TriRoute improves task success by 0.78 points over validator retry while cutting recovery-episode calls by 12.4%, wall-clock by 20.0%, tokens by 24.7%, and Azure cost by 24.9%. The paired interval is $[+0.30, +1.30]$, with corrected randomization $p = 0.0023$. Greedy TriEdit is tied with validator retry, so the 0.76-point routed-versus-greedy estimate supplies the same-library attribution; the remaining rows show that the policy advantage coexists with competitive generic and schema-aware alternatives.

A task-specialized upper context pairs a domain-aware router with a handwritten edit menu and reaches 80.21% at 4.89 calls (Appendix A12). That row quantifies the additional gain available when

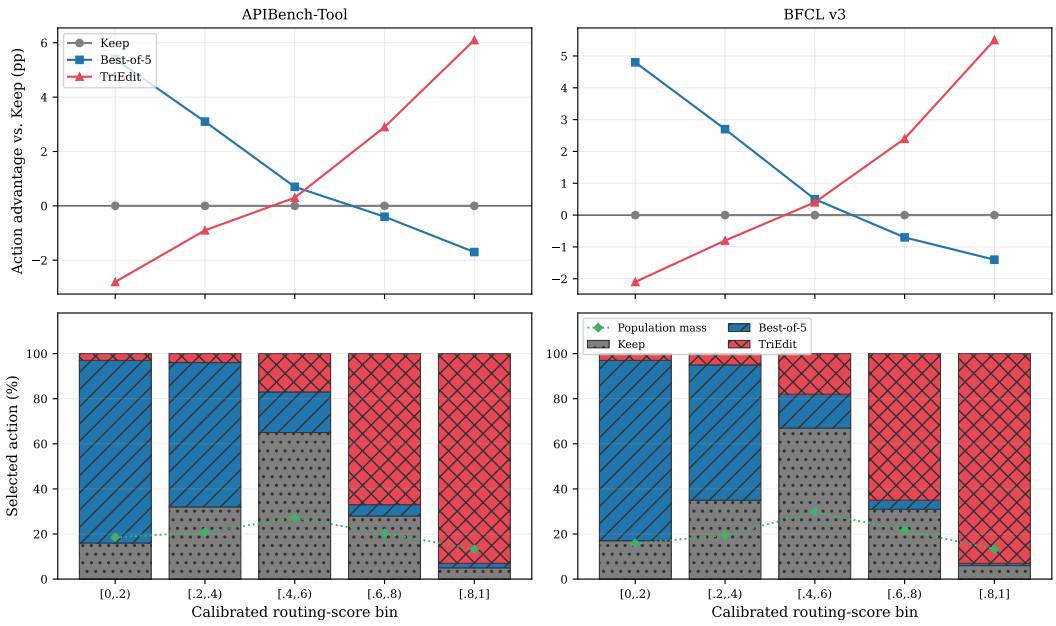

Figure 1: Failure-state routing on APIBench-Tool and BFCL v3. Top: descriptive per-bin action advantage relative to Keep. Bottom: modal instance-level TriRoute action among recovery-eligible attempts, with score-bin population mass overlaid; hatching makes actions distinguishable without color. Each instance is binned by its mean score over failed initial attempts, whereas thresholds act on individual attempts. *Binwise advantages are descriptive; the aggregate paired tests in Section 5.2 are the inferential endpoints.*

both routing information and repair capability are specialized. TriRoute supplies the corresponding frozen-policy result under an automatically mined fixed menu.

5.5 Failure-State Geometry and Learned Triage

Figure 1 exposes the action-value geometry aligned with TriRoute's frozen score. Resampling has its largest advantage on low-score diffuse failures, both additional actions have small point-estimate advantages in the low-separation center, and local editing becomes decisive on syntax-valid high-score near misses. The frozen validation objective assigns most center instances to Keep; after transfer from MBPP-Plus to tool calls, the held-out center means diagnose separation rather than retune the policy. APIBench-Tool and BFCL differ in population mass and transition sharpness yet preserve the low-score resample and high-score edit ordering. Appendix A10 gives the aggregation rule, counts, modal actions, success advantages, calls, and failure slices behind the curves.

Threshold heuristics recover part of the available gain, but the six-feature rule achieves the highest non-oracle success on every benchmark and uses fewer calls than the validation-tuned heuristics. Its intervals against the strongest heuristic exclude zero on APIBench-Tool, API-Bank, and NexusRaven; the BFCL estimate is $+0.46$ points with interval $[-0.04, +0.97]$. Across the suite, TriRoute's 25–29% modal edit mass, lower regret, and lower call use distinguish selective assignment from unconditional repair.

6 Discussion

The same-library contrast changes how recovery improvements should be attributed. Greedy TriEdit and validator retry are already competitive, yet the routed policy adds 0.73–0.76 points while reducing recovery-episode calls. Relative to greedy, that is roughly 7–8 additional successes per 1,000 requests and 230–250 fewer verifier evaluations per 1,000 recovery episodes; relative to validator retry on BFCL and API-Bank, the corresponding ranges are 10–12 additional successes and 690–710 fewer

evaluations. The domain-aware upper context further separates portable failure-state routing from the additional value of task-specific operators.

The demonstrated regime has structured tool calls, executable schema or semantic checks, a safe Keep decision, and a compact local-edit neighborhood. Within that regime, paired evidence supports the aggregate routed-policy effect, while score bins provide a descriptive mechanism view rather than independently significant subgroups. HotpotQA-Adv marks the complementary regime: when verifier feedback scores final correctness without exposing a local argument neighborhood, TriRoute and Best-of-8 are separated by only $+0.08 \pm 0.36$ points. The same-library estimand includes the policy's changed action frequencies, stopping behavior, resampling frequency, and realized calls; a matched-marginal randomization would be required to decompose state conditioning from those jointly chosen consequences. Likewise, $B{=}6$ controls verifier evaluations rather than generator FLOPs, and token, latency, and dollar comparisons are specific to APIBench-Tool. These boundaries make the result actionable: use failure-state routing when verifier predicates expose an executable local neighborhood, and interpret its gain as the value of the complete routed policy under that accounting contract.

7 Conclusion

Tool-call recovery benefits from separating action choice from action depth. On BFCL v3, API-Bank, and APIBench-Tool, TriRoute improves over schema-aware validator retry, and the identical-menu control estimates a 0.73–0.76 point gain over unconditional editing while using fewer recovery-episode evaluations. Score-stratified action values, cross-model transfer, validation-tuned heuristics, and operator-library interventions connect that aggregate gain to a stable low-score resample and high-score edit ordering.

For verifier-rich structured calls, inference-time scaling can allocate across recovery operators rather than only increasing samples or search steps within one operator. Under the evaluated six-evaluation ceiling, TriRoute turns the diagnostic structure already produced by verification into higher success with lower mean recovery demand.

Reproducibility Statement

Sections 3–5 define the action menu, call ledger, routing rule, local search, models, benchmarks, seeds, metrics, comparators, and inferential unit. Appendices A2–A10 provide operator preconditions, split construction, cross-model outcomes, robustness conditions, and request-level aggregation details; Appendix A12 describes comparator execution rules.

AI Use Statement

Generative AI tools were used only for manuscript writing, editing, and presentation. The authors reviewed all edits and remain responsible for the final content.

REFERENCES

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. doi: 10.48550/arXiv.2108.07732. URL https://arxiv.org/abs/2108.07732.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024. doi: 10.48550/arXiv.2407.21787. URL https://arxiv.org/abs/2407.21787.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024a. URL https://openreview.net/forum?id=cSimKw5p6R.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *International Conference on Learning Representations*, pp. 8746–8825, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/2460396f2d0d421885997dd1612ac56b-Abstract-Conference.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured nlp tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10932–10952. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.674. URL https://aclanthology.org/2023.emnlp-main.674/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783. URL https://arxiv.org/abs/2407.21783.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1321–1330, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Qijia He, Jiayi Cheng, Chenqian Le, Rui Wang, Xunmei Liu, Yixian Chen, Jie Mei, Zhihao Wang, Xupeng Chen, Yuhuan Chen, and Tao Wang. CodeRescue: Budget-calibrated recovery routing for coding agents. *arXiv preprint arXiv:2607.19338*, 2026. doi: 10.48550/arXiv.2607.19338. URL https://arxiv.org/abs/2607.19338v1. Version 1; withdrawn July 30, 2026 because submission preceded required company review and approval.

Qirui Jiao, Dian Jiao, Nan Du, Ying Shen, and Liang Lin. Invocation refiner: A plug-and-play module for rectifying LLM tool invocations. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14866–14898. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.731. URL https://aclanthology.org/2026.findings-acl.731/.

Claire Le Goues, Michael Pradel, and Abhik Roychoudhury. Automated program repair. *Communications of the ACM*, 62(12):56–65, 2019. doi: 10.1145/3318162. URL https://doi.org/10.1145/3318162.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3102–3116, 2023. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572, 2023. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Nexusflow.ai team. NexusRaven-V2: Surpassing GPT-4 for zero-shot function calling, 2023. URL https://huggingface.co/Nexusflow/NexusRaven-V2-13B. Model card; released December 5, 2023.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation? In *International Conference on Learning Representations*, pp. 36545–36593, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/9ddc141bdbf9d1db510cefff56c586ad-Abstract-Conference.html.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*, pp. 34433–34448, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5503a7c69d48a2f86fc00b3dc09de686-Abstract-Conference.html.

OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. doi: 10.48550/arXiv.2410.21276. URL https://arxiv.org/abs/2410.21276.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 48371–48392, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html.

John R. Rice. The algorithm selection problem. In *Advances in Computers*, volume 15, pp. 65–118. Elsevier, 1976. doi: 10.1016/s0065-2458(08)60520-3. URL https://doi.org/10.1016/S0065-2458(08)60520-3.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9895–9901, 2021. doi: 10.18653/v1/2021.emnlp-main.779. URL https://aclanthology.org/2021.emnlp-main.779/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, pp. 10131–10165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. RestGPT: Connecting large language models with real-world RESTful apis. *arXiv preprint arXiv:2306.06624*, 2023. doi: 10.48550/arXiv.2306.06624. URL https://arxiv.org/abs/2306.06624.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Chunqiu Steven Xia, Yuxiang Wei, and Lingming Zhang. Automated program repair in the era of large pre-trained language models. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering*, pp. 1482–1494, 2023. doi: 10.1109/icse48619.2023.00129. URL https://doi.org/10.1109/ICSE48619.2023.00129.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying LLM-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE):801–824, 2025. doi: 10.1145/3715754. URL https://doi.org/10.1145/3715754.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822, 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, pp. 9965–10017, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pp. 62138–62160, 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

A1 Accounting and Heuristic Summary

The following tables provide the action ledger used in Section 3 and the condensed heuristic comparison discussed in Section 5. The ledger includes the initial evaluation; the heuristic comparison separates measured success from oracle regret.

Table 5: Verifier-evaluation ledger. The hard ceiling includes the cached initial evaluation. Resource means condition on recovery-eligible attempts with imperfect y_0; success rates include every attempt.

Case or action	Charged evaluations	Hard maximum
Perfect initial candidate	1	1
Keep	1	1
Best-of-5	$1 + 4$ fresh evaluations	5
TriEdit	$1 + N_{\text{edit}},\ 0 \le N_{\text{edit}} \le K$	6
Validator retry	$1 + N_{\text{retry}},\ 0 \le N_{\text{retry}} \le 5$	6

The hard ceiling applies to each attempt, while resource averages condition on imperfect initial candidates. This distinction is shared by all action rows.

Table 6: TriRoute versus the strongest validation-tuned heuristic, the two-feature syntax-validity plus failure-count rule. Success averages all attempts, calls average recovery episodes, and regret is the oracle-minus-TriRoute success gap in percentage points. Appendix A8 reports every heuristic and action fraction.

Benchmark	Heuristic	TriRoute	Oracle	$\Delta_{\text{TriRoute}-H}$ [95% CI]	Regret
APIBench-Tool	78.42/4.99	79.34/4.93	80.07	+0.92 [+0.24,+1.54]	0.73
BFCL v3	81.13/4.97	81.59/4.91	82.18	+0.46 [−0.04,+0.97]	0.59
API-Bank	74.65/4.95	75.23/4.89	75.92	+0.58 [+0.07,+1.09]	0.69
NexusRaven	73.92/4.98	74.38/4.92	75.02	+0.46 [+0.01,+0.91]	0.64

The strongest two-feature heuristic is competitive on BFCL v3; the remaining three benchmark intervals exclude zero. Full heuristic rows appear in Appendix A8.

A2 Operator Catalog

The headline experiments use the same five TriEdit operators on APIBench-Tool, BFCL v3, API-Bank, τ-bench, and NexusRaven v2. `add-required-field` inserts a missing required argument from the schema or the nearest validated prompt literal. `fix-type-coerce` repairs simple type mismatches such as stringified integers or booleans. `remove-extra-field` deletes arguments that violate the schema, while `fix-enum-value` projects a near-enum string onto the nearest admissible value. `normalize-path` canonicalizes URL or path arguments through case, slash, and percent-encoding normalization. Router fitting on MBPP-Plus instantiates the same local-search procedure with its pre-registered AST edit menu to produce local-edit-optimal labels. Cross-domain reuse transfers the six-feature scoring rule and thresholds; the five operators above remain the fixed edit capability in every tool-call result.

Each operator is guarded by explicit preconditions. The schema must parse successfully before an edit is proposed, and any edited candidate that remains invalid, duplicate, or off-schema is discarded before a verifier call is charged. ToolBench validation diffs are canonicalized by touched schema path and edit type before clustering, and we keep only templates with support at least four and mean

verifier gain at least 0.08 on the held-out validation pool. Templates containing benchmark-specific tool identifiers or task literals are dropped. This construction keeps selective use of a small fixed menu as the controlled experimental variable and excludes test-time menu growth from the attribution.

Algorithm 1 TriRoute rolling local edit search with cached initial verification

Require: Candidate y_0, cached $v_0 = V(y_0)$, verifier V, operators $\mathcal{O}$, post-initial edit limit K, radius r
Ensure: Improved candidate $y^\star$

Set $y^\star \leftarrow y_0$, $\text{score}^\star \leftarrow \|v_0\|_1$, and edit count $c \leftarrow 0$
Initialize $\alpha_o, \beta_o \leftarrow 1$ for each $o \in \mathcal{O}$
while $c < K$ and an admissible unseen composition remains **do**
  Sample $\theta_o \sim \text{Beta}(\alpha_o, \beta_o)$ for each o
  From incumbent $y^\star$, compose compatible sequence σ of at most r highest-sampled operators; set $y' \leftarrow \sigma(y^\star)$
  if y' is valid and unseen **then**
    Evaluate $s' \leftarrow \|V(y')\|_1$; set $c \leftarrow c + 1$
    Update each $o \in \sigma$ with success indicator $\mathbb{1}[s' > \text{score}^\star]$
    if $s' = m$ **then**
      return y'
    end if
    if $s' > \text{score}^\star$ **then**
      $y^\star \leftarrow y'$ and $\text{score}^\star \leftarrow s'$
    end if
  else
    Mark σ unavailable and increment β_o for each $o \in \sigma$
  end if
end while
return $y^\star$

The pseudocode makes the cached initial charge, early success return, and rolling-incumbent expansion explicit. Each proposal has radius at most r from its current incumbent, rather than from the original candidate. The same search procedure is invoked by both routed and greedy policies.

A3 Training and Evaluation Splits

Table 7: Development sources used to freeze the router and tool-call operator library, followed by the evaluated tool-call benchmarks.

Role	Source	Split unit	Instances	Attempts	Use
Router fitting	MBPP-Plus	problem	250	2,000	coefficients and thresholds
Menu construction	ToolBench validation	tool class	30	480	feature adapter and five templates
Controlled test	APIBench-Tool	instance	384	3,840	held-out tool classes
External tests	BFCL v3 / API-Bank	instance	638 / 753	6,380 / 7,530	primary outcomes
Transfer tests	NexusRaven / τ-bench	instance	318 / 165	3,180 / 1,650	frozen-policy transfer

Table 7 separates every source that can affect the policy from the benchmarks used for claims. MBPP-Plus supplies the logistic coefficients and joint thresholds; disjoint ToolBench validation classes define the tool-call feature adapter and frozen five-template menu. The split unit is a complete problem, tool class, or benchmark instance, and all attempts from one unit remain together during fitting and uncertainty estimation. APIBench-Tool holds out tool classes, while BFCL v3, API-Bank, NexusRaven, and τ-bench enter only after the router, thresholds, and menu are frozen.

A3.1 APIBENCH-TOOL

APIBench-Tool is the controlled JSON/tool-call study built from ToolBench (Qin et al., 2024). We select instances whose reference answer invokes exactly one of 42 pre-registered tools with JSON Schema Draft 2020-12 specifications and semantic post-condition checks. The test set contains 384 instances spread evenly across 12 held-out tool classes, while 30 disjoint validation tool classes provide the only tool-call data used for feature-adapter development and operator mining. Verification first applies strict schema validation and then evaluates semantic post-conditions over a mock tool registry. This design fixes tool inventory, verifier semantics, held-out classes, and accounting in one controlled intervention.

A3.2 VERIFIER ADAPTERS

Every benchmark adapter emits the same ordered predicate types: parse validity, tool-name validity, required-field presence, argument type and enum validity, extra-field or path validity, execution status when available, and benchmark-native semantic post-conditions. Unsupported predicates are omitted rather than imputed, so pass rate and failed-check count use the adapter-specific m. Parse failure sets syntax validity to zero and fails all predicates that require a parsed argument object; otherwise syntax validity is one. Fail clusters share a schema path or native post-condition identifier. The official benchmark checker remains the final success endpoint and is not replaced by the routing score.

Table 8: Benchmark-to-pass-vector contract. All adapters preserve native task success while exposing executable diagnostics to the frozen six-feature map.

Benchmark	Recovery-time predicates	Final success endpoint
APIBench-Tool	JSON parse; schema path, required, type, enum, and extra-field checks; mock-registry post-conditions	Schema-valid arguments and all semantic post-conditions
BFCL v3	Official parse, function, argument, execution, and post-condition verdicts converted to binary predicates	Official BFCL instance success
API-Bank	Official invocation parse, API and argument checks, execution status, and semantic completion verdicts	Official API-Bank task success
NexusRaven v2	Function-call parse, selected function, argument schema, and reference-value checks	Official invocation match
τ-bench	Emitted-call parse and argument checks plus native environment-state post-conditions	Native task completion after returning the selected call to the interaction loop

A4 EXTERNALLY AUTHORED BENCHMARK PROVENANCE

BFCL v3 (Patil et al., 2025) and API-Bank (Li et al., 2023) anchor the primary evidence beyond the ToolBench-derived APIBench-Tool study; both reuse the same six-feature router, thresholds, and five auto-mined operators without benchmark-specific additions. NexusRaven v2 and τ-bench extend the test across independent function-calling and multi-turn tool-interaction settings. The predeclared HotpotQA-Adv control (Yang et al., 2018) tests a distinct verifier geometry and yields no material gain when the signal does not expose a locally repairable neighborhood.

A5 τ-BENCH SUPPORT CHECK

Table 9: Predeclared τ-bench support check under the unchanged frozen policy. Calls and Azure cost average recovery episodes.

Policy (B=6 unless noted)	Success (%)	Δ vs. ours	Calls	Azure $/episode	Role
Best-of-8 (uncapped ref.)	72.03 ± 0.53	-3.11	8.00	\$0.0195	cost anchor
Best-of-5 (shared ceiling)	71.87 ± 0.55	-3.27	5.00	\$0.0131	fixed resampling
Agentless-Repair	73.58 ± 0.51	-1.56	5.12	\$0.0128	generic repair
OpenAI strict:true	72.84 ± 0.50	-2.30	4.97	\$0.0119	schema-aware
Outlines-0.0.46 GBNF	73.22 ± 0.49	-1.92	4.98	\$0.0117	schema-aware
Validator-feedback retry	74.29 ± 0.48	-0.85	5.04	\$0.0125	strongest schema-aware
TriRoute (frozen)	75.14 ± 0.45	—	4.91	*\$0.0082*	routed policy

The protocol, baseline list, thresholds, and April 15, 2026 evaluation specification were fixed before the τ-bench run. Strict structured output and Outlines each charge the cached initial verification plus at most four constrained candidates, terminating the schedule when a candidate verifies perfectly; their recovery-episode means can therefore fall below five. TriRoute exceeds every shared-ceiling comparator, leading validator retry by 0.85 points while reducing recovery-episode verifier calls and Azure cost. The result extends the same decision consequence to an independently authored multi-turn tool benchmark without changing the router or edit menu.

A6 ADDITIONAL OUT-OF-FAMILY CONFIRMATION: NEXUSRAVEN V2

Table 10: Additional out-of-family confirmation on NexusRaven v2 under the unchanged frozen policy. Calls average recovery episodes.

Method	Success (%)	Diff. (pp)	95% CI	Rand. p	Calls
Best-of-8 (uncapped ref.)	71.84 ± 0.55	-2.54	$[-3.17, -1.91]$	0.00002	8.00
Best-of-5 (shared ceiling)	71.61 ± 0.57	-2.77	$[-3.37, -2.17]$	0.00001	5.00
Agentless-Repair	72.95 ± 0.53	-1.43	$[-2.04, -0.82]$	0.0006	5.70
Strict function calling	72.52 ± 0.52	-1.86	$[-2.48, -1.24]$	0.00008	5.00
Grammar-constrained decoding	72.81 ± 0.51	-1.57	$[-2.18, -0.96]$	0.0004	5.00
Validator-feedback retry	73.28 ± 0.52	-1.10	$[-1.72, -0.48]$	0.0012	5.61
Same-library greedy TriEdit	73.67 ± 0.50	-0.71	$[-1.16, -0.26]$	0.0024	5.16
TriRoute (frozen)	74.38 ± 0.49	—	—	—	4.92

The unchanged router, thresholds, operator menu, backend, and accounting produce 74.38% success on NexusRaven. TriRoute leads validator retry by 1.10 points and same-library greedy editing by 0.71 while lowering recovery-episode calls, reproducing the accuracy–efficiency direction outside the ToolBench, BFCL, and API-Bank lineages.

A7 CROSS-MODEL BASELINES AND TRANSFER CRITERIA

The following tables use the same public instances, ten seeds, prompts, verifier adapters, thresholds $(0.30, 0.70)$, five-template menu, and B=6 verifier ceiling. Each cell is all-attempt success percentage / mean recovery-episode calls. Outlines is the constrained-decoding baseline available across all three generators.

Table 11: Cross-model method details for GPT-4o.

Method	Metric	BFCL (n=638)	API-Bank (n=753)	NexusRaven (n=318)
Best-of-5	success / calls	78.68/5.00	72.11/5.00	71.61/5.00
Outlines-0.0.46	success / calls	80.11/5.00	73.71/5.00	72.81/5.00
Validator retry	success / calls	80.54/5.62	74.02/5.58	73.28/5.61
Same-library greedy	success / calls	80.86/5.14	74.47/5.12	73.67/5.16
TriRoute	success / calls	81.59/4.91	75.23/4.89	74.38/4.92

On GPT-4o, TriRoute leads validator retry and same-library greedy repair on all three external benchmarks while using fewer recovery-episode evaluations.

Table 12: Cross-model method details for Llama-3.1-70B-Instruct.

Method	Metric	BFCL (n=638)	API-Bank (n=753)	NexusRaven (n=318)
Best-of-5	success / calls	72.66/5.00	65.98/5.00	63.89/5.00
Outlines-0.0.46	success / calls	73.22/5.00	66.93/5.00	64.92/5.00
Validator retry	success / calls	73.97/5.61	67.55/5.57	65.56/5.59
Same-library greedy	success / calls	74.19/5.17	67.84/5.13	65.91/5.15
TriRoute	success / calls	74.82/4.98	68.41/4.96	66.72/4.99

The frozen policy preserves positive margins for Llama-3.1-70B across BFCL, API-Bank, and NexusRaven; the same-library API-Bank interval is the one Llama cell that includes zero.

Table 13: Cross-model method details for DeepSeek-R1-Distill-Qwen-32B.

Method	Metric	BFCL (n=638)	API-Bank (n=753)	NexusRaven (n=318)
Best-of-5	success / calls	73.81/5.00	67.62/5.00	65.02/5.00
Outlines-0.0.46	success / calls	74.74/5.00	68.51/5.00	66.48/5.00
Validator retry	success / calls	75.52/5.60	69.20/5.55	67.57/5.58
Same-library greedy	success / calls	75.79/5.15	69.52/5.12	67.86/5.14
TriRoute	success / calls	76.28/4.95	70.16/4.94	68.37/4.97

DeepSeek-32B also preserves positive point estimates throughout, with same-library intervals excluding zero on API-Bank and including zero on BFCL and NexusRaven.

Table 14: Paired outcomes and frozen-router transfer diagnostics. W/T/L compares TriRoute with validator retry over $n\times10$ attempts. p_V/p_G are randomization tests versus validator/greedy. Precision and AUROC use the local-edit-optimal target; regret is in percentage points.

Model	Benchmark	W/T/L vs. V	p_V/p_G	Precision	AUROC	Regret
GPT-4o	BFCL	402/5643/335	0.0008/0.0016	82.4	0.889	0.59
	API-Bank	478/6665/387	0.0003/0.0014	81.6	0.876	0.69
	NexusRaven	225/2765/190	0.0012/0.0024	79.8	0.864	0.64
Llama-70B	BFCL	381/5672/327	0.0048/0.025	80.7	0.873	0.88
	API-Bank	426/6743/361	0.009/0.061	79.1	0.861	1.02
	NexusRaven	252/2713/215	0.003/0.024	78.3	0.856	1.11
DeepSeek-32B	BFCL	365/5698/317	0.011/0.081	81.2	0.881	0.82
	API-Bank	448/6706/376	0.004/0.034	79.9	0.868	0.94
	NexusRaven	226/2753/201	0.028/0.120	77.8	0.852	1.19

The predeclared router-quality criteria are positive mean gain over validator retry, TriEdit-selection precision of at least 75%, AUROC of at least 0.85, and oracle regret below 1.5 points. Every model–benchmark cell satisfies these four criteria without generator-specific fitting; same-library randomization tests are separately reported in Table 14.

A8 FULL THREE-ACTION HEURISTIC ABLATION

Heuristic thresholds are selected on the same validation split and with the same search budget used for TriRoute, then frozen before test. K/B/R gives the percentage of benchmark instances whose modal action across recovery-eligible seeded attempts is Keep, Best-of-5, or TriEdit. Attempt-level actions remain deterministic functions of each failed attempt's score; W/T/L gives listed-policy wins / ties / TriRoute wins over all paired instance-seed outcomes.

Table 15: Three-action ablation on APIBench-Tool.

Policy	Success/calls	K/B/R (%)	Regret	Δ [95% CI]	W/T/L
Always Keep	70.31/1.00	100/0/0	9.76	$-9.03[-10.30, -7.80]$	114/3265/461
Always Best-of-5	76.22/5.00	0/100/0	3.85	$-3.12[-3.80, -2.50]$	175/3370/295
Always greedy TriEdit	78.58/5.18	0/0/100	1.49	$-0.76[-1.21, -0.31]$	222/3367/251
Pass-rate threshold	77.86/5.21	21/54/25	2.21	$-1.48[-2.10, -0.87]$	195/3393/252
Distance-to-perfect	78.11/5.10	23/51/26	1.96	$-1.23[-1.84, -0.62]$	207/3379/254
Syntax + failure count	78.42/4.99	25/50/25	1.65	$-0.92[-1.54, -0.24]$	214/3377/249
TriRoute	79.34/4.93	27/48/25	0.73	—	—
Oracle action	80.07/4.88	29/44/27	0	$+0.73[+0.30, +1.18]$	248/3372/220

On APIBench-Tool, the six-feature score gains 0.92 points over the strongest two-feature heuristic and closes the policy-to-oracle gap to 0.73 points.

Table 16: Three-action ablation on BFCL v3.

Policy	Success/calls	K/B/R (%)	Regret	Δ [95% CI]	W/T/L
Always Keep	73.42/1.00	100/0/0	8.76	$-8.17[-9.19, -7.14]$	173/5513/694
Always Best-of-5	78.68/5.00	0/100/0	3.50	$-2.91[-3.48, -2.34]$	276/5642/462
Always greedy TriEdit	80.86/5.14	0/0/100	1.32	$-0.73[-1.18, -0.28]$	291/5751/338
Pass-rate threshold	80.72/5.09	24/50/26	1.46	$-0.87[-1.43, -0.31]$	283/5758/339
Distance-to-perfect	80.94/5.03	25/48/27	1.24	$-0.65[-1.20, -0.11]$	298/5743/339
Syntax + failure count	81.13/4.97	27/46/27	1.05	$-0.46[-0.97, +0.04]$	304/5743/333
TriRoute	81.59/4.91	29/43/28	0.59	—	—
Oracle action	82.18/4.86	31/40/29	0	$+0.59[+0.14, +1.04]$	347/5724/309

On BFCL v3, TriRoute has the highest non-oracle success and uses fewer calls than the validation-tuned heuristics; its $+0.46$ point margin over the strongest heuristic has interval $[-0.04, +0.97]$.

Table 17: Three-action ablation on API-Bank.

Policy	Success/calls	K/B/R (%)	Regret	Δ [95% CI]	W/T/L
Always Keep	66.18/1.00	100/0/0	9.74	$-9.05[-10.11, -8.00]$	218/6412/900
Always Best-of-5	72.11/5.00	0/100/0	3.81	$-3.12[-3.73, -2.51]$	331/6633/566
Always greedy TriEdit	74.47/5.12	0/0/100	1.45	$-0.76[-1.23, -0.29]$	352/6769/409
Pass-rate threshold	74.12/5.07	23/51/26	1.80	$-1.11[-1.68, -0.54]$	336/6774/420
Distance-to-perfect	74.38/5.01	25/49/26	1.54	$-0.85[-1.40, -0.30]$	348/6770/412
Syntax + failure count	74.65/4.95	27/47/26	1.27	$-0.58[-1.09, -0.07]$	361/6764/405
TriRoute	75.23/4.89	30/44/26	0.69	—	—
Oracle action	75.92/4.84	32/41/27	0	$+0.69[+0.22, +1.16]$	422/6738/370

On API-Bank, the six-feature partition improves on the strongest heuristic by 0.58 points and reduces oracle regret to 0.69 points.

Table 18: Three-action ablation on NexusRaven v2.

Policy	Success/calls	K/B/R (%)	Regret	Δ [95% CI]	W/T/L
Always Keep	64.37/1.00	100/0/0	10.65	−10.01[−11.44, −8.58]	94/2674/412
Always Best-of-5	71.61/5.00	0/100/0	3.41	−2.77[−3.37, −2.17]	151/2790/239
Always greedy TriEdit	73.67/5.16	0/0/100	1.35	−0.71[−1.16, −0.26]	159/2839/182
Pass-rate threshold	73.41/5.10	22/52/26	1.61	−0.97[−1.58, −0.36]	151/2847/182
Distance-to-perfect	73.68/5.04	24/50/26	1.34	−0.70[−1.27, −0.13]	163/2832/185
Syntax + failure count	73.92/4.98	26/47/27	1.10	−0.46[−0.91, −0.01]	169/2827/184
TriRoute	74.38/4.92	28/45/27	0.64	—	—
Oracle action	75.02/4.87	30/42/28	0	+0.64[+0.19, +1.09]	198/2804/178

On NexusRaven, TriRoute retains a $+0.46$ point margin over the strongest heuristic and a 0.64 point oracle gap under the unchanged thresholds.

A9 Operator-Library Robustness

The frequency-ranked three-template menu keeps the most common edit families and sets r=2. The held-out-source menu contains five templates mined from disjoint tool classes and removes benchmark-specific literals. Within each row, greedy and routed policies use the identical menu.

Table 19: Operator-library robustness under B=6. Coverage is the share of initial failures matching at least one operator precondition; off-menu is the share requiring an edit outside the library. Success/calls report all-attempt success and recovery-episode calls for greedy and TriRoute.

Benchmark	Library	Coverage	Off-menu	Greedy	TriRoute	Δ [95% CI]
BFCL	Frozen five	82.7	11.6	80.86/5.14	81.59/4.91	+0.73 [+0.28, +1.18]
	Pruned three	68.3	23.9	79.92/5.08	80.71/4.88	+0.79[+0.26, +1.32]
	Held-out source	74.5	18.7	80.34/5.12	81.01/4.94	+0.67[+0.12, +1.21]
API-Bank	Frozen five	79.4	14.2	74.47/5.12	75.23/4.89	+0.76 [+0.29, +1.23]
	Pruned three	64.7	27.6	73.41/5.06	74.10/4.87	+0.69[+0.16, +1.22]
	Held-out source	71.2	22.1	73.88/5.10	74.51/4.91	+0.63[+0.08, +1.19]
NexusRaven	Frozen five	76.8	17.1	73.67/5.16	74.38/4.92	+0.71 [+0.26, +1.16]
	Pruned three	61.9	30.8	72.61/5.09	73.22/4.90	+0.61[+0.04, +1.18]
	Held-out source	69.6	24.9	72.98/5.13	73.54/4.94	+0.56[−0.03, +1.15]

Reducing operator coverage lowers absolute success and increases off-menu failures, but routed-minus-greedy remains positive in every condition. The effect varies from 0.56 to 0.79 points, with eight of nine intervals excluding zero; the held-out-source NexusRaven estimate has the same direction with a wider interval. The fixed-menu policy effect therefore persists across materially different repair capabilities rather than depending on one optimally shaped library.

A10 Failure-Mode Mechanism and Request-Level Counts

The five visualization bins are fixed before test. Each benchmark instance is placed in one bin by its mean calibrated score over recovery-eligible seeded attempts, so the Count column sums to unique instances that exhibit at least one initial failure; every displayed benchmark instance does so. Thresholds act separately on each failed attempt; K/B/R therefore reports the modal recovery action, and the displayed call mean is computed over recovery episodes rather than reconstructed from modal labels. Action advantages are descriptive percentage-point success differences relative to Keep within each bin.

Table 20: Instance-binned descriptive data behind Figure 1. Count is unique instances with at least one failed initial attempt; K/B/R is the modal recovery action; calls average recovery episodes.

Benchmark	Score bin	Count	Best-of-5 advantage	TriEdit advantage	K/B/R (%)	Calls
APIBench-Tool	$[0, .2)$	71	$+5.4$	-2.8	$16/81/3$	5.14
	$[.2, .4)$	80	$+3.1$	-0.9	$32/64/4$	5.08
	$[.4, .6)$	104	$+0.7$	$+0.3$	$65/18/17$	3.92
	$[.6, .8)$	78	-0.4	$+2.9$	$28/5/67$	4.74
	$[.8, 1]$	51	-1.7	$+6.1$	$5/2/93$	4.91
BFCL v3	$[0, .2)$	101	$+4.8$	-2.1	$17/80/3$	5.12
	$[.2, .4)$	124	$+2.7$	-0.8	$35/60/5$	5.05
	$[.4, .6)$	190	$+0.5$	$+0.4$	$67/15/18$	3.79
	$[.6, .8)$	138	-0.7	$+2.4$	$31/4/65$	4.68
	$[.8, 1]$	85	-1.4	$+5.5$	$6/1/93$	4.89

Table 21: APIBench-Tool failure slices. Before is initial success; after is TriRoute success. Selected action reports the modal action and its within-slice frequency.

Failure slice	Denominator	Before	After	Selected action	Calls
Missing field	82	61.2	88.9	TriEdit 63%	4.84
Type mismatch	76	64.7	87.5	TriEdit 59%	4.79
Enum mismatch	51	68.1	85.7	TriEdit 57%	4.76
Extra field or path	67	66.9	84.2	TriEdit 54%	4.73
Schema-valid semantic	64	72.4	79.8	Keep 41%	4.18
Non-local or off-menu	44	42.7	58.6	Best-of-5 71%	5.02

The mechanism analysis explains the aggregate policy effect without treating the displayed bins as independent hypothesis tests. Resampling has the largest measured advantage in the lowest bins, while editing has the largest advantage in the highest; the center has small positive point estimates for both paid actions and the validation-frozen policy most often stops. Missing-field, type, enum, and path failures concentrate TriEdit selections and large absolute recovery, while non-local failures concentrate resampling. Aggregate comparisons use paired instance bootstrap and randomization tests and remain the inferential endpoint.

A11 FAILURE-STATE TRIAGE MATRIX

Table 22: Failure-state triage matrix for executable tool-call arguments. Each row links verifier-visible structure to the frozen policy action, a diagnostic comparator, and manuscript evidence.

Failure state	Verifier-visible evidence	Policy action	Diagnostic comparator	Held-out or stress setting	Outcome and manuscript trace
Diffuse low-score failure	Many distributed violations or invalid syntax	Best-of-5	Always resample / validator retry	Lowest fixed score bins on APIBench-Tool and BFCL	Resampling advantage $+5.4$ and $+4.8$ points; Table 20
Low-separation state	Similar conditional action values	Keep	Always Keep and threshold heuristics	Middle fixed score bins on both benchmarks	Keep is the modal instance action for 65% and 67%; Table 20
Concentrated schema fault	Missing, type, enum, or path violation	TriEdit	Same-library greedy TriEdit	Held-out APIBench tool classes	After-triage success spans 84.2–88.9%; Table 21
Schema-valid semantic fault	Valid object with low action separation	Keep / selective resample	Validator-feedback retry	Semantic post-condition slice	79.8% success at 4.18 calls; Table 21
Non-local or off-menu fault	No admissible local transformation	Best-of-5	Greedy TriEdit	Non-local slice and pruned / held-out menus	Best-of-5 is modal at 71%; Tables 21 and 19
Generator and tool shift	Frozen features under a changed error mix	Three-way triage	Validator retry and same-library greedy	Three generators and held-out public tools	Positive mean validator and greedy margins in every cross-model cell; Tables 3 and 14

The matrix unifies the mechanism evidence as a decision map. Its rows connect the policy's transition from resampling through stopping to editing with increasingly concentrated and executable verifier structure. The cross-model and operator-library interventions then test the stability of the low-score and high-score ordering under generator and repair-coverage shifts.

A12 COMPLETE COMPARATOR TABLES

All capped comparators charge and cache the initial evaluation. A perfect initial candidate exits before recovery and contributes to success but not to resource means. For each recovery episode, Best-of-5 evaluates all four independent samples after the cached candidate, then returns the first successful candidate in generation order, or otherwise the highest-scoring candidate with generation-order tie breaking. Validator-feedback retry appends the currently failed verifier predicates to the correction prompt and stops on success or after six total evaluations. Strict structured output and Outlines each evaluate at most four constrained candidates after the cached initial evaluation and terminate on a verifier-perfect candidate; their conditional means can consequently be below five. Agentless-Repair receives the same initial failure predicates, proposes localized argument edits, and stops on success or at the shared ceiling. Greedy TriEdit invokes Algorithm 1 on every imperfect initial candidate; TriRoute replaces that unconditional decision with the frozen three-action rule.

Table 23: Full GPT-4o external comparator set. Success is mean percentage ± one standard error over all attempts; calls average recovery episodes. All rows except Best-of-8 satisfy $B \leq 6$.

Method	BFCL success	BFCL calls	API-Bank success	API-Bank calls
Best-of-8 (uncapped)	78.92 ± 0.47	8.00	72.41 ± 0.52	8.00
Best-of-5	78.68 ± 0.49	5.00	72.11 ± 0.54	5.00
Agentless-Repair	79.78 ± 0.51	5.71	73.55 ± 0.50	5.68
Strict structured output	79.85 ± 0.45	5.00	73.25 ± 0.49	5.00
Outlines-0.0.46	80.11 ± 0.44	5.00	73.71 ± 0.48	5.00
Validator-feedback retry	80.54 ± 0.45	5.62	74.02 ± 0.49	5.58
Same-library greedy TriEdit	80.86 ± 0.44	5.14	74.47 ± 0.48	5.12
TriRoute	81.59 ± 0.42	4.91	75.23 ± 0.47	4.89

Across both external benchmarks, TriRoute improves success over shared-ceiling Best-of-5, uncapped Best-of-8, schema-aware generation, validator retry, and same-library greedy editing. The paired effects in Table 1 quantify the complete routed policy's gain over unconditional use of the identical edit library.

Table 24: Additional APIBench-Tool comparators. The domain-aware row changes both routing information and the edit menu and is reported as task-specialized upper context, not as a same-library control. Tokens and USD are totals over 384 instances.

Method	Success	Calls	Wall	Tokens (k total)	USD total
Self-Debug + resampling	77.11 ± 0.67	5.70	5.05	347.6	3.39
Reflexion + Search	76.83 ± 0.64	5.40	4.95	338.7	3.30
LATS-Edit	77.22 ± 0.59	5.30	4.85	332.9	3.24
Tree-of-Thoughts Edit	77.41 ± 0.62	5.20	4.80	328.0	3.19
Domain-aware router + handwritten menu	80.21 ± 0.48	4.89	4.20	276.0	2.69

A13 HotpotQA-Adv Boundary Test

The HotpotQA-derived HotpotQA-Adv control (Yang et al., 2018) tests a verifier regime in which the available signal scores final answer correctness but does not identify a compact local argument-edit neighborhood. The predeclared protocol uses 480 adversarial examples, the unchanged router and thresholds, ten seeds, and up to five post-initial edit evaluations. Best-of-8 reaches 52.38% and TriRoute reaches 52.46%, for a realized difference of $+0.08 \pm 0.36$ points. The result bounds the mechanism claim: when verifier feedback does not separate locally repairable failures, the policy does not produce a material routing gain.